A Study of Adaptive Forward Error Correction for Wireless Collaborative Computing

Philip K. McKinley, *Member*, *IEEE*, Chiping Tang, and Arun P. Mani

Abstract—This paper addresses the problem of reliably multicasting Web resources across wireless local area networks (WLANs) in support of collaborative computing applications. An adaptive forward error correction (FEC) protocol is described, which adjusts the level of redundancy in the data stream in response to packet loss conditions. The proposed protocol is intended for use on a proxy server that supports mobile users on a WLAN. The software architecture of the proxy service and the operation of the adaptive FEC protocol are described. The performance of the protocol is evaluated using both experimentation on a mobile computing testbed as well as simulation. The results of the performance study show that the protocol can quickly accommodate worsening channel characteristics in order to reduce delay and increase throughput for reliable multicast channels.

Index Terms—Collaborative computing, wireless local area networks, forward error correction, adaptive middleware, reliable multicast, object-oriented software design.

1 Introduction

The large-scale deployment of wireless communication services and advances in portable computers are quickly making ubiquitous computing into a reality. One class of applications that can benefit from this expanding and varied infrastructure is collaborative computing. Examples include computer-supported cooperative work, computer-based instruction, and mobile operator support in industrial installations. Given their synchronous and interactive nature, collaborative applications are particularly sensitive to heterogeneity among the computing devices and the network connections used by participants. Pavilion [1] is an object-oriented framework for developing collaborative Web-based applications. Pavilion provides a suite of proxy-based services and protocols that help to mitigate differences among networks and devices.

This paper describes a study of reliable multicasting across wireless local area networks (WLANs) to support collaborative applications. The main contribution of this work is a new reliable multicast protocol, implemented as a Pavilion proxy service, that dynamically adjusts the level of forward error correction (FEC) in response to channel loss behavior. The protocol is based on block erasure codes [2], [3], due to their ability to correct uncorrelated packet losses among multiple mobile receivers. The results of a performance study, involving both experimentation and simulation, demonstrate that the protocol can quickly accommodate dynamic channel characteristics in order to reduce delay and increase throughput.

The remainder of the paper is organized as follows: Section 2 provides background information on the Pavilion middleware framework and its operation. Section 3 discusses the relevant issues in reliable multicasting across WLANs. Section 4 describes the architecture and implementation of the FEC proxy, including details of the new multicast protocol. Sections 5 and 6, respectively, present the results of an experimental study and a simulation study of the protocol. Related work is discussed in Section 7, and Section 8 summarizes the results and discusses future directions.

2 Pavilion Framework and the WBRM Protocol

This section presents the context for this study by reviewing the design and operation of the Pavilion framework. Pavilion is written in Java™ and supports collaboration using off-the-shelf browsers such as Netscape Navigator and Microsoft Internet Explorer. In default mode, Pavilion operates as a collaborative Web browser, as depicted in Fig. 1a. A member of the group acquires leadership through the leadership protocol. On the leader's system, shown on the left in Fig. 1a, the browser interface monitors the activities of the Web browser. The interface is notified whenever a new URL is loaded by the browser, and it reliably multicasts this URL to all other participants. The Web resource itself and any embedded/linked files are reliably multicast by the leader's proxy server to the proxy servers of the other group members. At each receiving system, the browser interface requests the local Web browser to load the new URL. The target Web browser will subsequently initiate retrieval of the files, via its proxy, which will return the requested items. While browsing, the collaborating users can speak with each other through real-time audio channels [4].

In addition to supporting collaborative browsing, Pavilion components can be reused and extended in order to construct new collaborative applications. For example, Pavilion has been used to develop VGuide [5], a collaborative virtual

- *P.K. McKinley and C. Tang are with the Department of Computer Science and Engineering, Michigan State University, East Lansing MI 48824. E-mail: {mckinley, tangchip}@cse.msu.edu.*
- *A.P. Mani is with Lucent Technologies-Bell Laboratories, 101 Crawfords Corner Rd., Homdel, NJ 07733. E-mail: amani@lucent.com.*

Manuscript received 10 June 2001; accepted 8 Apr. 2002.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 116283.

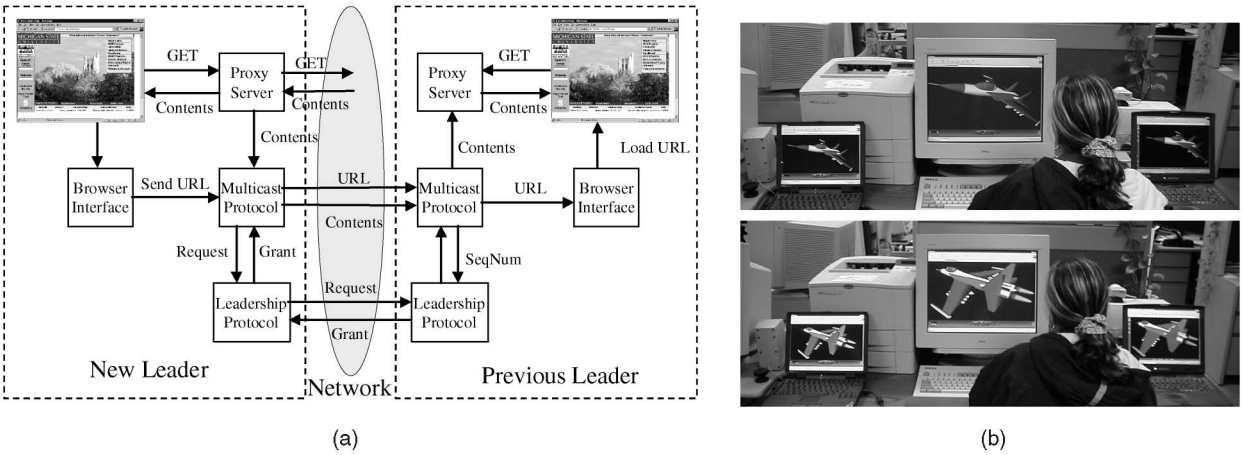

Fig. 1. Operation and application of the Pavilion framework. (a) Default operation of Pavilion as a collaborative browsing tool. (b) a VGuide user demonstrates synchronous navigation of a VRML world on a desktop and two wireless laptops.

reality application. VGuide enables a user to select a VRML (Virtual Reality Modeling Language) file from the Internet and lead a group of users through that virtual world. Fig. 1b shows two photographs of VGuide running on a wired desktop workstation and two wireless laptop computers.

Reliable multicasting in Pavilion is provided by the Web-Based Reliable Multicast (WBRM) protocol [6], an application-level protocol that implements reliability atop UDP/IP multicast. The WBRM protocol is a receiver-initiated, or NAK-based, protocol: a receiver notifies the sender only when it misses a packet in the stream [7]. Fig. 2 shows the WBRM protocol architecture. Both the sending and receiving components of the protocol comprise a set of Java threads and data structures. Flow Control in WBRM is extensible and user configurable. The default method is rate-based and is similar to that of the RAMP protocol [8]: the delay between packets is a function of the ratio of the total number of packets sent during an interval to the number of NAKs received during the same interval. Details of the operation and performance of the WBRM protocol on *wired* networks can be found in [6], [1]. The study presented in this paper, investigates how to provide better support for wireless hosts.

3 Issues in Wireless Reliable Multicast

Reliable multicast services for wireless channels must address several challenging problems. First, the packet loss rates are highly dynamic and location-dependent [9]. For example, Fig. 3a shows the results of a short excursion, using a laptop computer, in the wireless testbed used in this study. Moving away from the wireless access point can quickly produce low signal-to-noise (SNR) ratios and high packet loss rates. Even near the access point, the performance of reliable multicasting can suffer if flow control is not handled properly. Fig. 3b shows the results of a set of experiments where the WBRM protocol was used to reach multiple wireless receivers, all with high SNR values. In these tests, the interpacket delay values used in the rate-based flow control method were fixed. While the protocol

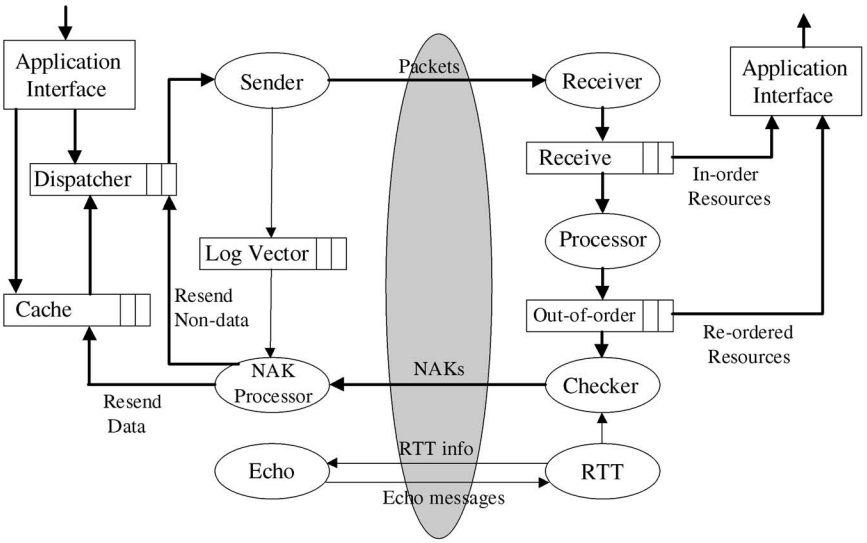

Fig. 2. WBRM protocol architecture. The sender maintains a log vector describing the resource stream and a cache of recently transmitted resources. The receiver sends NAKs for missing packets, while buffering those that arrive intact but out-of-order.

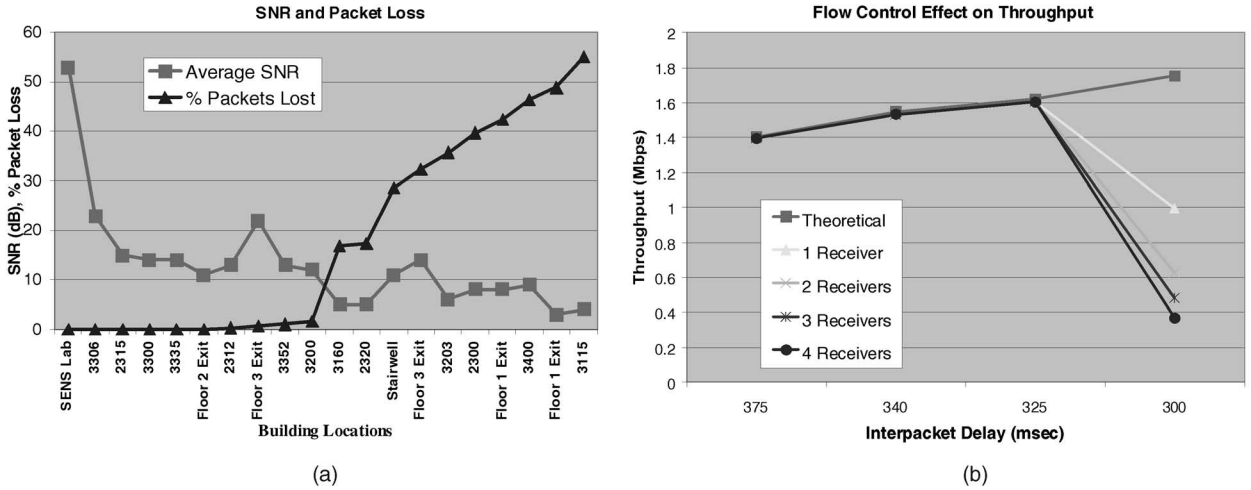

Fig. 3. Experimental measurements of wireless LAN performance. (a) SNR values and packet loss rate. (b) Flow control effect on reliable multicast throughput.

can achieve a throughput of approximately 1.6 Mbps on the 2 Mbps link,[1] the value drops dramatically when the sending rate is too high, due to buffer overflow at the wireless access point.

Second, the loss characteristics of a WLAN are very different from that of a wired network. In a wired domain, losses occur mainly due to congestion, which can be addressed with flow control. In wireless domain, however, losses are often due to external factors such as interference and antennae alignment. Fig. 4 shows a sample of the burst error behavior, gathered from two locations in the testbed. Location 1 is just outside the room containing the access point, and Location 2 is approximately 10 meters down a hallway. While some large bursts occur, the majority are under three packets long, and most "burst" errors comprise a single packet loss. Such results are encouraging because they imply that a relatively small amount of redundant information in the data stream might correct most errors, with retransmissions necessary only for long burst errors, which are less common.

Third, the 802.11b CSMA/CA MAC layer provides RTS/CTS signaling and link-level acknowledgements for unicast frames, but not for multicast frames [10]. The result is a higher packet loss rate as observed by applications using UDP/IP multicast. Fig. 5 demonstrates this behavior by showing typical traces of packet delivery rates for Location 2. The combination of RTS/CTS and link-level retransmissions provide a lossless unicast channel, while the multicast channel experiences 5 to 15 percent packet losses.

Finally, since the wireless channel is a shared broadcast medium, it is important to minimize the amount of feedback from receivers. Simultaneous responses from multiple receivers can cause channel congestion and burden the access point, thereby hindering the forward transmission of data frames. Thus, it is desirable to minimize the number of NAKs sent by receivers.

1. All performance results presented in this paper are for a 2 Mbps WaveLAN network. We are currently conducting follow-on studies using an 11 Mbps Cisco/Aironet WLAN.

4 Proxy Architecture and Operation

To address the lower bandwidth and higher error rates of wireless channels, we constructed a Pavilion proxy service that lies between the wireless nodes and the rest of the wired network [9]. Fig. 6 shows the physical configuration of the proxy in serving three mobile hosts. The proxy executes two instances of the WBRM protocol, one for the wired network and the other for the wireless network, enabling the WBRM flow control algorithm to tune itself independently for each network segment. Moreover, any requests for retransmissions by the wireless receivers can be handled by the proxy which, due to its proximity to the receivers, can usually respond more quickly than the original data source.

The overall throughput of the multicast connection is limited by the bandwidth of the wireless channel. However, the typical mismatch in speeds between the wired and wireless networks means that the proxy may have to perform temporary buffering for large files. Since the Pavilion proxy is typically a dedicated workstation, the in-memory WBRM cache can be quite large (several megabytes), and it automatically grows into secondary storage, as needed.

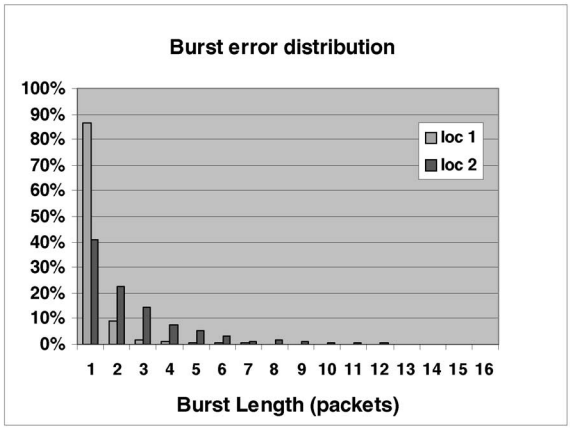

Fig. 4. Typical WLAN packet loss characteristics.

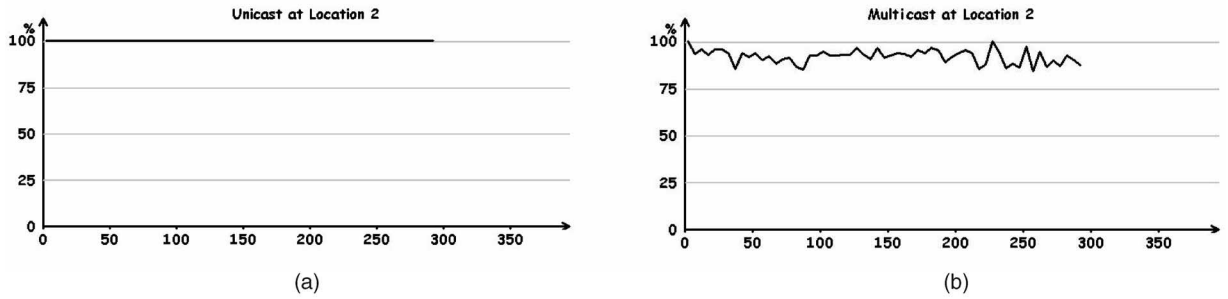

Fig. 5. Unicast and multicast packet loss traces. (a) UDP unicast packet reception rate. (b) UDP multicast packet reception rate.

4.1 Forward Error Correction

One way to address the high loss rates of wireless channels is to insert redundant information into the data stream, enabling a receiver to correct some losses without contacting the sender for retransmission. This study focuses on *erasures* of packets resulting from CRC-based detection of errors at the data link layer. As shown in Fig. 7, an (n, k) *block erasure code* [2] converts k source packets into n encoded packets, such that any k of the n encoded packets can be used to reconstruct the k source packets. These codes have gained popularity recently due to an efficient implementation by Rizzo [11]. Each set of n encoded packets is referred to as a *group*. This study uses only *systematic* (n, k) codes, meaning that the first k packets in a group are identical to the original k data packets. The remaining $n - k$ packets are referred to as *parity* packets.

The advantage of using block erasure codes for reliable multicasting is that a single parity packet can be used to correct independent single-packet losses among different receivers [11]. Hence, sending parity packets with data packets reduces the number of NAKs sent by receivers. Moreover, when receivers do require additional parity packets, the sender can respond to NAKs from different receivers with a single set of parity packets, rather than a different set for each receiver.

4.2 Proxy Architecture

Fig. 8 shows the design of the Pavilion FEC proxy. For clarity, only those components used in the wired-to-wireless direction are shown. Most of the proxy components are written in Java and comprise one or more threads. The lone exception is the FEC Encoder, which uses Rizzo's public domain C implementation [3]. With minor modifications, this code is invoked from the proxy code using the Java Native Interface [12].

Several proxy components are reused directly from the original WBRM protocol, while other components (shaded) were designed for this study. The *WBRM Receiver* is reused without modification. It receives multicast data packets over the wired network and delivers a reliable output stream of these packets to the *Packet Buffer*. The remaining components implement the sending half of the proxy, which we refer to as the *wireless* WBRM (W-WBRM) protocol. The *FEC Group Filter* collects the data packets into FEC data blocks of size k and places them in the Dispatcher Queue. The *FEC Encoder* monitors the Dispatcher Queue. When it detects that a group of k packets is full, it invokes the encoding routines and produces the $n - k$ parity packets. Depending on the current proactive rate, discussed below, some number of these parity packets are placed in the Dispatcher Queue to be sent proactively with the data packets. The remaining parity packets are stored and used to respond to NAKs from receivers. The *Packet Dispatcher* is reused from WBRM and simply uses UDP/IP multicast to transmit the packets in the Dispatcher Queue. If a receiver detects that it has received fewer than k packets from a particular group, then it informs the proxy by sending a NAK message (under the constraints of local and global NAK suppression, which are discussed later). Based on received NAKs, the *NAK Processor* signals the Dispatcher to send additional parity packets. If the parity packets for a group are exhausted, then the protocol will begin retransmitting the parity packets. If $n < 2k$, the protocol will retransmit data packets as well. In all the experiments reported here, $n \geq 2k$.

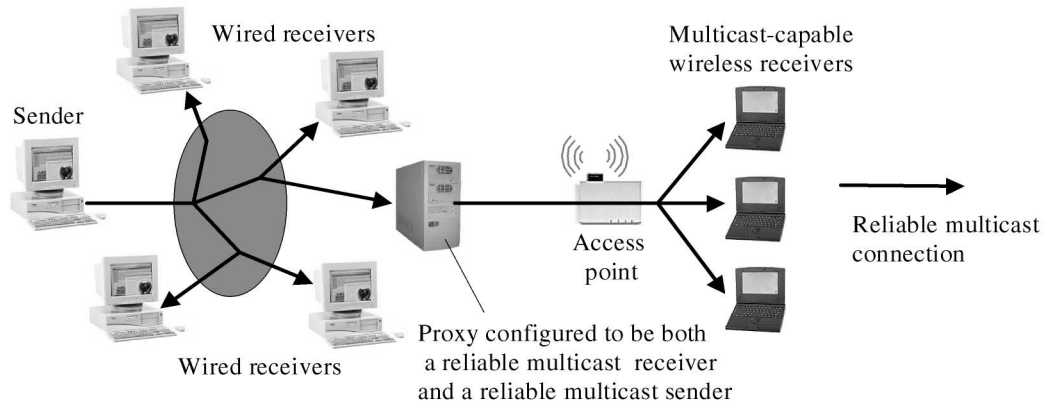

Fig. 6. Proxy configuration for nodes on a wireless LAN.

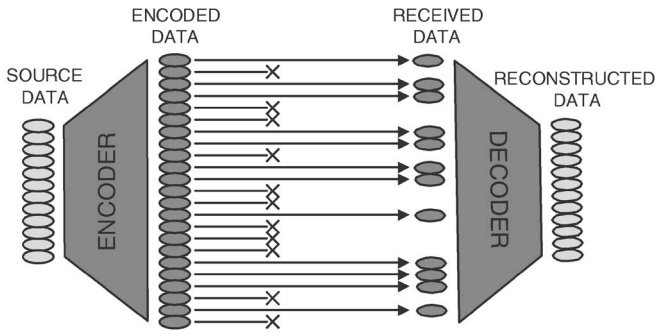

Fig. 7. Operation of FEC based on block erasure codes.

4.3 Need for Adaptive FEC

For a given packet group, after sending the k data packets, the W-WBRM protocol could send all $n-k$ parity packets immediately. However, in many situations, not all of these packets will be needed, and their transmission will consume bandwidth unnecessarily. Instead, the level of redundancy is determined by the value of a *proactive parameter*, α. For each group of n encoded packets, the proxy immediately sends $\lceil k(1+\alpha) \rceil$ packets (the ceiling function is ignored in the remaining discussion). Any receiver that loses fewer than αk of these packets can recover from the losses locally, while a receiver that loses more than αk packets will send a NAK to the proxy requesting additional parity packets. The NAK format includes fields to identify G, the packet group, and L, the number of packets required by this receiver to reconstruct the original k data packets in group G. It is important to emphasize that in the W-WBRM protocol α is applied to *all* transmissions, including transmission of requested parity packets. Hence, in response to a NAK, the NAK Processor makes available to the Dispatcher $L(1+\alpha)$ additional parity packets for transmission.

To study the effects of the different α values on the performance of the protocol for different loss rates, a set of experiments was conducted; a sampling of the results is shown in Fig. 9. The plots show the results for reliably multicasting to three wireless laptop receivers under various loss rates: 5 percent, 10 percent, and 20 percent. In order to control the error rate, random packet losses are emulated on all the hosts (experimental loss conditions are discussed later). An FEC packet size of $1,400$ bytes was used to transfer a 4 MB file to wireless receivers. The FEC parameters (n,k) were $(60,20)$, that is, 40 parity packets are computed for each group of 20 data packets. Hence, if $\alpha = 1.0$, then 20 data packets are sent, followed by 20 proactive parity packets. The remaining 20 parity packets reside at the proxy and are used to respond to NAKs.

Fig. 9 demonstrates two important results about the throughput for the wireless receivers. First, as expected, the ideal value for α varies with network conditions. Specifically, the α value at which throughput is maximized, increases with the packet loss rate. Therefore, the W-WBRM protocol should update the value of α in response to changing loss rates among receivers. Second, all the curves follow a similar pattern: a relatively steep ascent prior to the optimal value, and a gradual descent beyond this value. This behavior suggests that sending more parity packets than is necessary is better than relying on feedback in the form of NAKs from receivers. However, sending too many unneeded parity packets will eventually reduce throughput.

Given these observations, the W-WBRM protocol adjusts the value of α dynamically in response to packet loss behavior. As shown in Fig. 8, a plug-in component to the NAK Processor called the *Packet Loss Monitor* collects information on NAKs and forwards it to the *Adaptive FEC Control* plug-in, which adjusts the value of the α accordingly. Two functions, α_{inc} and α_{dec}, are used to increase and decrease α, respectively. The next two sections describe a series of experiments and simulations that attempt to determine which definitions of these functions are most effective, to which values their parameters should be set, and how they interact with other aspects of the W-WBRM protocol.

5 Experimental Performance Evaluation

To evaluate the performance of the W-WBRM protocol, experiments were conducted on a mobile computing testbed. The testbed comprises conventional workstations

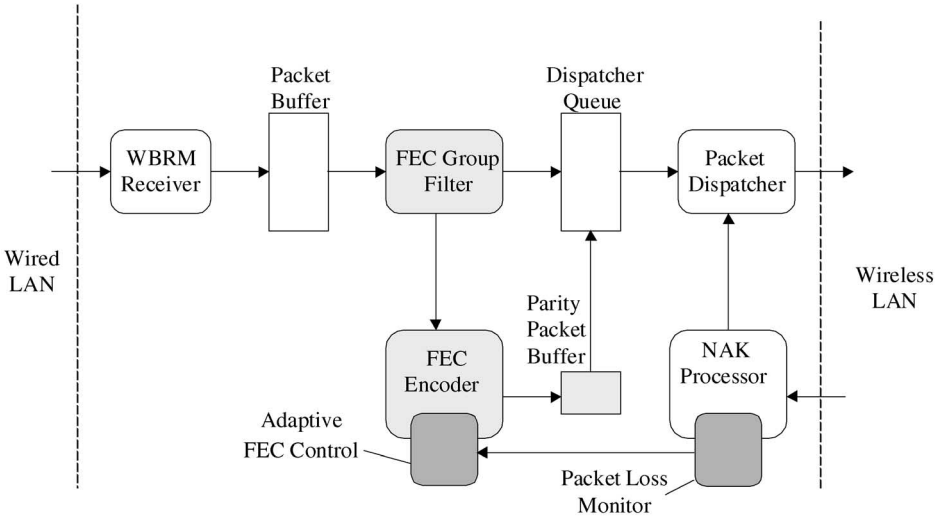

Fig. 8. Operation of FEC proxy and the W-WBRM protocol.

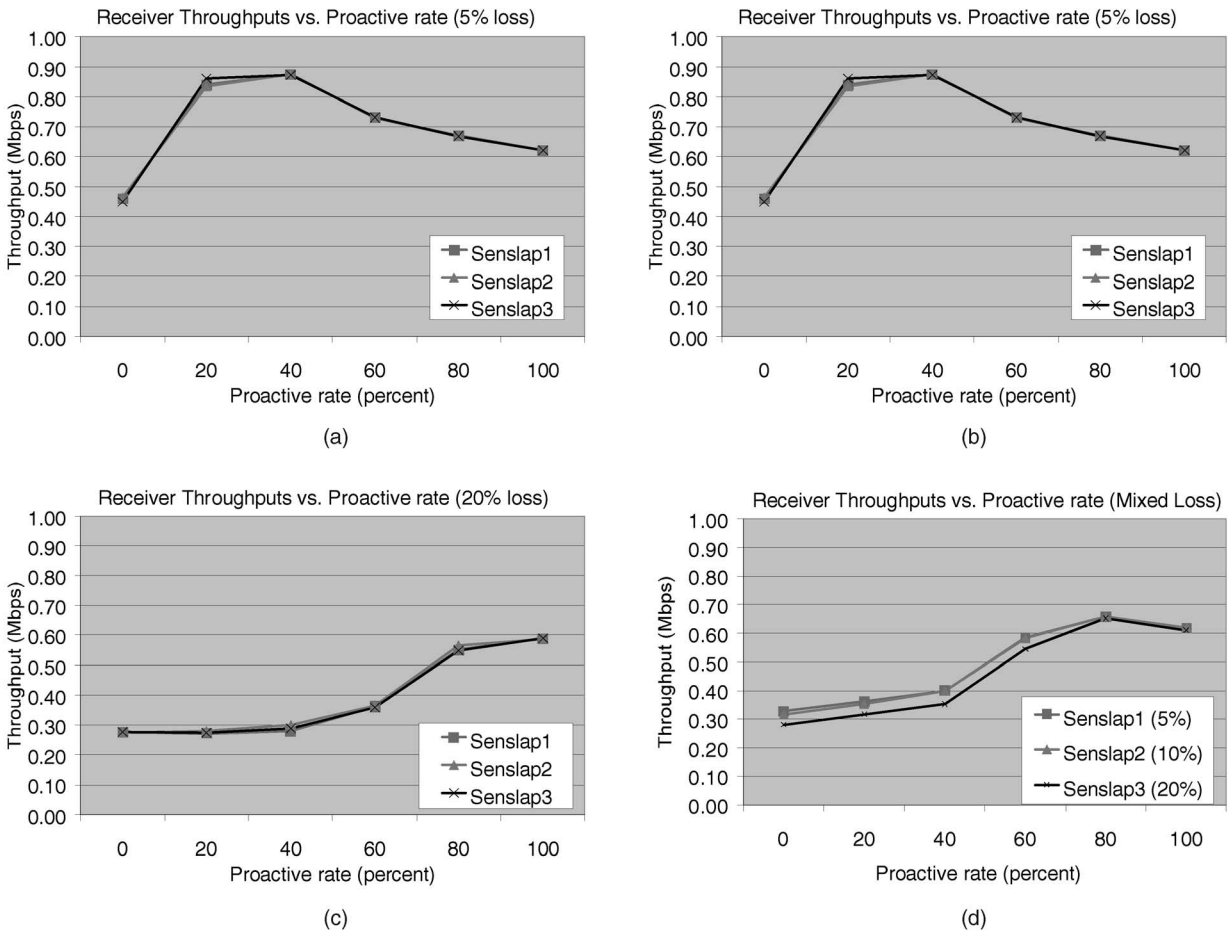

Fig. 9. Throughput results with static proactive rates. (a) 5 percent packet loss at all receivers. (b) 10 percent packet loss at all receivers. (c) 20 percent packet loss at all receivers. (d) Mixed packet loss at receivers.

connected by a 100 Mbps Fast Ethernet switch, various WLANs (Lucent WaveLAN, Proxim RangeLAN2, and Cisco/Aironet), and several mobile computer systems. All tests reported here were conducted on the Lucent WaveLAN network, which uses direct sequence spread spectrum signaling and has a raw bit rate of 2 Mbps. The sender, proxy, and receivers were configured as shown in Fig. 6. The sender and proxy were executed on dual-processor 400/450 MHz desktop workstations, while the mobile nodes were 300 MHz laptops equipped with WaveLAN network interface cards. The WaveLAN access point and the participating wired stations were located in our laboratory, while the locations of the mobile nodes were varied.

Evaluating Adaptive Parameters. The first set of experiments is designed to evaluate the effects of different α_{inc} and α_{dec} functions on the behavior of the W-WBRM protocol. Initially, α is increased in an additive manner, that is, the adaptive FEC control plug-in sets $\alpha = \alpha + \alpha_{\text{inc}}$, where α_{inc} is based on observation of NAK behavior for each group. Specifically, $(\alpha_{\text{inc}} = M * L/k)$, where L is the requested number of parity packets, and M is a small integer (specific values are discussed later). This approach is intentionally conservative, since the results in Fig. 9 indicated that it is less costly to overestimate α than to underestimate it. Different receivers may request additional parity packets for the same group, so the NAK Processor collapses these requests by comparing the L values in NAKs for the same packet group that are received within a given window of time. This operation, called *parity packet suppression*, attempts to avoid both unnecessary transmission of parity packets and overly large increases in the value of α.

The α_{dec} function prevents α from remaining high, relative to the needed level of redundancy. If no NAKs have arrived at the proxy in a window of W groups, the value of α is reduced using the function α_{dec}. In this set of experiments, $\alpha_{\text{dec}} = 0.02$. Therefore, in the absence of NAKs, the value of α is reduced periodically by 2 percent. Eventually, α becomes low enough that the receivers produce one or more NAKs. At this point, α is again increased by an amount relative to the number of requested packets, and the process repeats. When used to support a Pavilion collaborative session, the objective is to keep the the number of proactive parity packets "hovering" slightly above the number actually needed to decode the data packets.

Fig. 10 shows a sample of the results, in which a 4MB file is sent repeatedly via the proxy to a single wireless receiver. The FEC parameters are $(60, 20)$ and as before, losses are emulated by dropping packets randomly. The packet size is 1,400 bytes and the mean packet loss rate in all cases is set to 20 percent. Each plot contains three curves that illustrate the behavior of the protocol as the experiment progresses. The *Required Parity* curve shows, for each group in the file, the

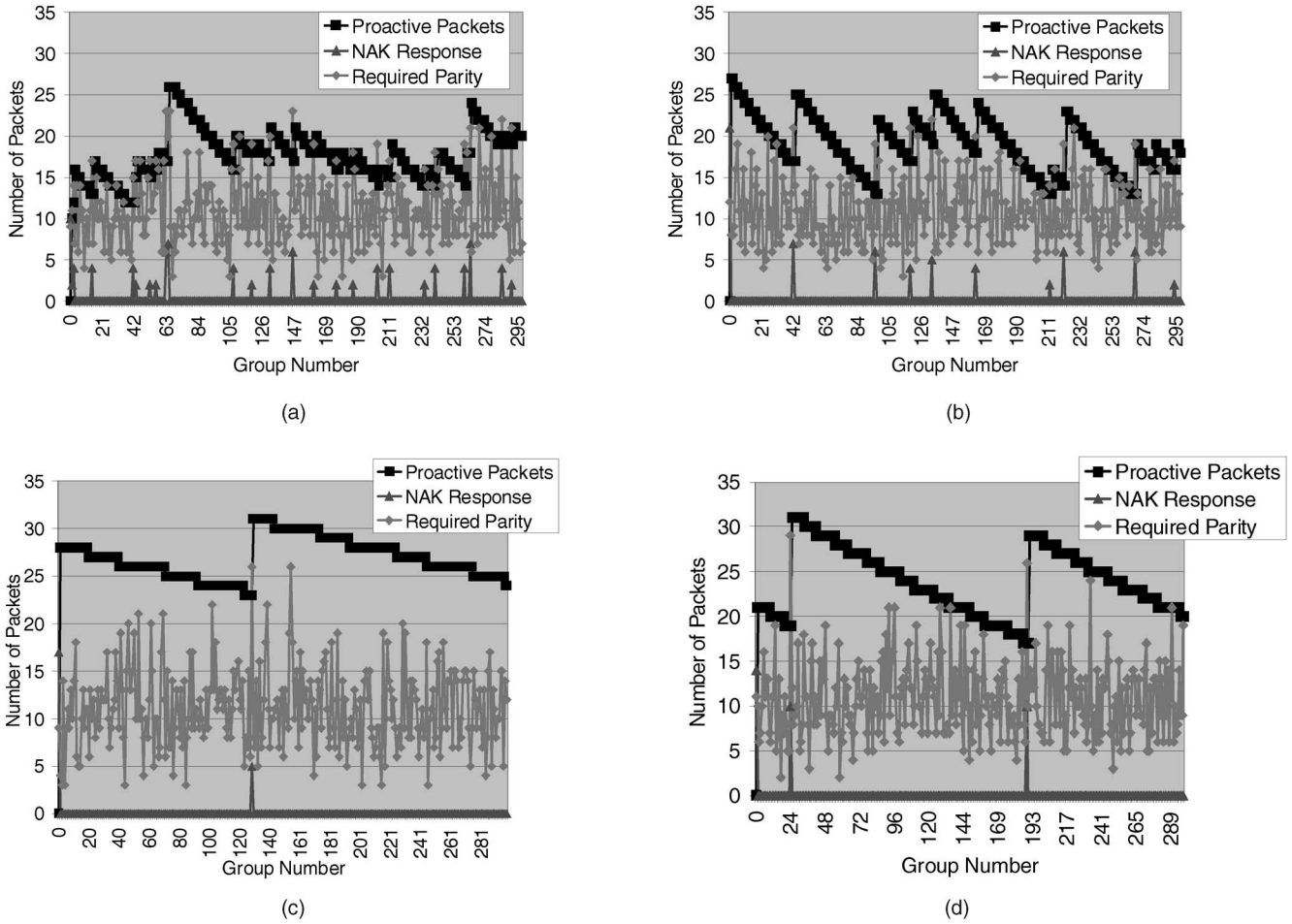

Fig. 10. Adaptive FEC behavior for single receiver, varying parameters M and W. (a) $\alpha_{\text{inc}} = 2 * L/k, W = 3$ groups. (b) $\alpha_{\text{inc}} = 3 * L/k, W = 3$ groups. (c) $\alpha_{\text{inc}} = 4 * L/k, W = 10$ groups. (d) $\alpha_{\text{inc}} = 3 * L/k, W = 10$ groups.

maximum number of parity packets needed by a receiver to decode the source packets of the group. The *Proactive Packets* curve shows, for each group, the total number of parity packets proactively sent by the proxy. The *NAK Response* curve in the graph shows the total number of parity packets actually sent by the proxy in response to (all) NAKs from the receivers. Whenever the number of proactive packets is less than the required number of parity packets, feedback is received in the form of a NAK (indicated by the spikes in *NAK Response* curve), and the proactive parameter α is bumped to a higher value.

Clearly, the values of parameters M and W directly affect the NAK behavior of the protocol. When $M = 2$ (that is, $\alpha_{\text{inc}} = 2 * L/k$), as shown in Fig. 10a, the proactive rate α is increased in response to a NAK, but the receivers often require additional parity packets in subsequent groups, producing many NAKs. Increasing M to 3 improves the situation, as shown in Fig. 10b, but with the window size W set to 3 groups, α decreases too quickly, producing many situations in which additional parity packets must be sent in response to NAKs. By increasing W to 10, as shown in Figs. 10c and 10d, the protocol limits this behavior. As shown in Fig. 10c, however, if the value of M is too large (4 in this case), the protocol may transmit too many unneeded parity packets. In the remainder of the experiments reported in this section, M is set to 3, although additional values are used in the simulations in Section 6.

Varying Loss Rate. Fig. 11 shows the behavior of the W-WBRM protocol when multicasting to three wireless laptop computers under different (emulated) packet loss conditions: 5 percent, 10 percent, and 20 percent. In these tests, $M = 3$ and the FEC parameters are $(60, 20)$. The value of α_{dec} is set to 0.02 and $W = 10$, so α will decrease by 0.02 every 10 groups (in the absence of NAKs). The protocol does a reasonable job of keeping the proactive rate above, but not too far above, the packet loss rate. However, since the update of α is based on instantaneous perceived loss rate corresponding to NAKs, rather than the average error rates, the result is occasionally higher or lower than the required value. For example, the graph for a 5 percent loss rate shows a initial surge in the proactive packets based on a relatively large loss suffered in the first packet group. Although the protocol eventually recovers from the over-estimation, one might be tempted to use a "smoothing function" of NAK response behavior when increasing α. However, given the earlier results indicating that an excess of parity packets is less detrimental to performance than a shortage thereof, the approach used here is to increase α quickly in response to negative conditions. Fig. 11d plots the resulting throughput when using adaptive FEC in the W-WBRM

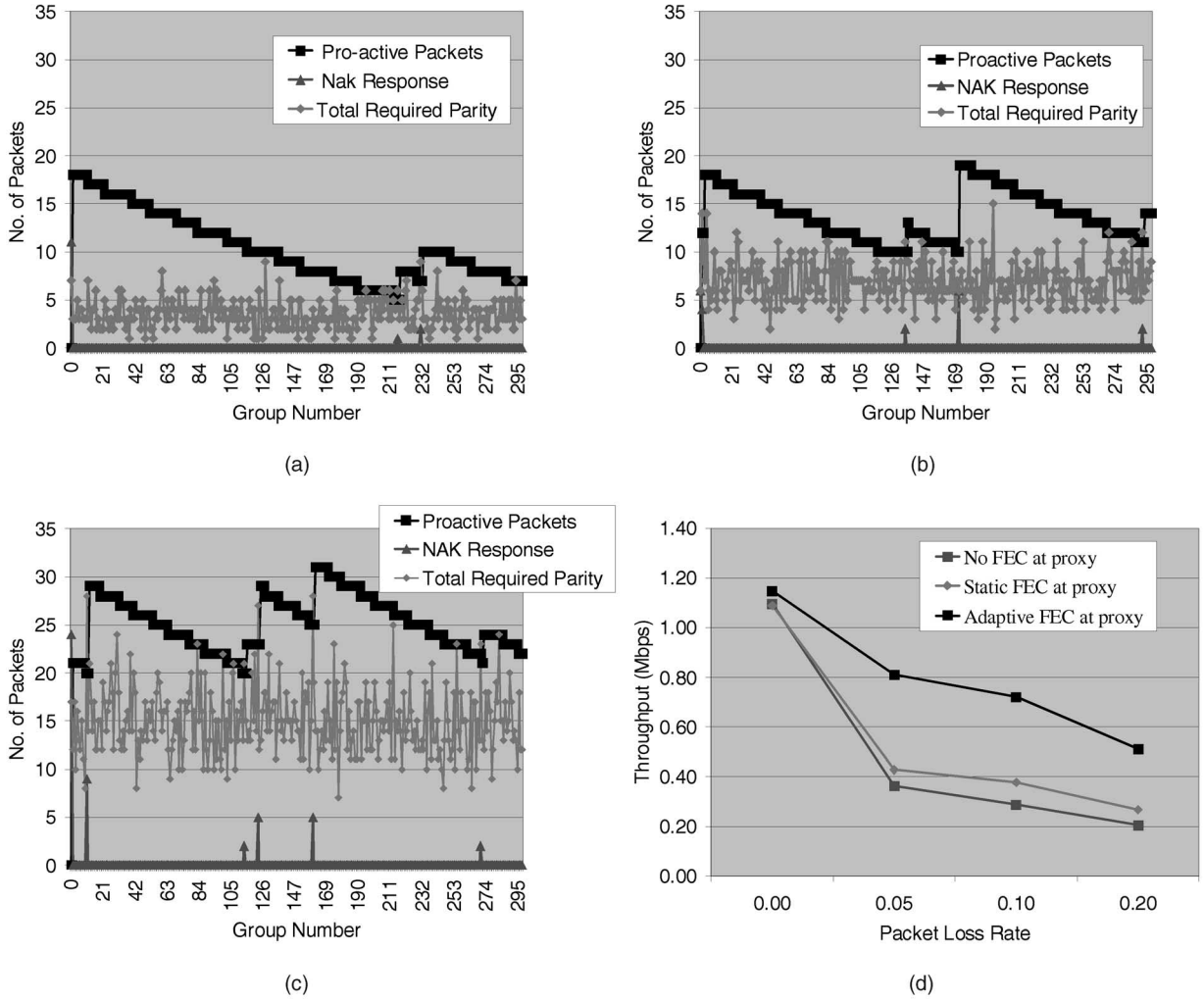

Fig. 11. Performance of adaptive FEC protocol under various loss conditions (three laptop receivers). (a) 5 percent packet loss at all recievers. (b) 10 percent packet loss at all receivers. (c) 20 percent packet loss at all receivers. (d) Measured throughputs.

protocol, compared to the original WBRM protocol (no FEC) and W-WBRM with a fixed-rate, or *static*, FEC. The adaptive protocol improves throughput by approximately a factor of 2 for all situations involving packet loss.

Handling Bursty Losses. In addition to random packet losses, the study also evaluated the W-WBRM protocol under bursty losses, which are common in wireless LANs. Designing α_{inc} and α_{dec} for burst error situations is probably most effective for small groups of receivers, where the aggregate loss patterns of the group are also very bursty. Fig. 12a shows an example of this behavior. A 4MB file was multicasted repeatedly to a laptop as it was moved within the area covered by the accesss point. The resulting packet loss behavior shown in Fig. 12a, as well as that in the other subfigures, exhibits a bifurcation between 1) relatively large burst errors and 2) random single-packet losses. For such environments, it might be desirable to decrease α faster than linear. Figs. 12b, 12c, and 12d show results for the W-WBRM protocol when dealing with real (as opposed to emulated) packet losses. Three laptop receivers were involved; two remained near the access point and one was carried by a user who traversed a nearby hallway. In Fig. 12b, $\alpha_{\text{dec}} = 0.02$, as in earlier tests. While the proactive parity packets handle most of the losses, the overhead is quite large. In Fig. 12c, α_{dec} is modified so that α decreases exponentially. In this case, the protocol correctly predicts several large losses and handles them. However, the lower bound on α is 0, and many single-packet losses produce NAKs. In Fig. 12d, the same exponential function is used, but the protocol always sends at least one proactive packet. In this case, the NAK feedback is reduced considerably.

6 Simulation Results

To explore configurations and conditions not available on the experimental testbed, a simulation study was conducted. The study used MX [13], a simulation tool that enables unmodified application programs to be executed atop a simulated network. In MX, a file written in a configuration description language describes the characteristics of the network components: host computers, routers, network interface cards, etc. A socket-level API enables application code to be linked with the simulator; both Java and C++ interfaces are currently supported. Protocol modules can be linked together to form a protocol stack; supported modules include TCP, UDP, IP, 802.3, and 802.11.

The MX simulator was used to determine whether and how the proposed methods might need to be tuned for

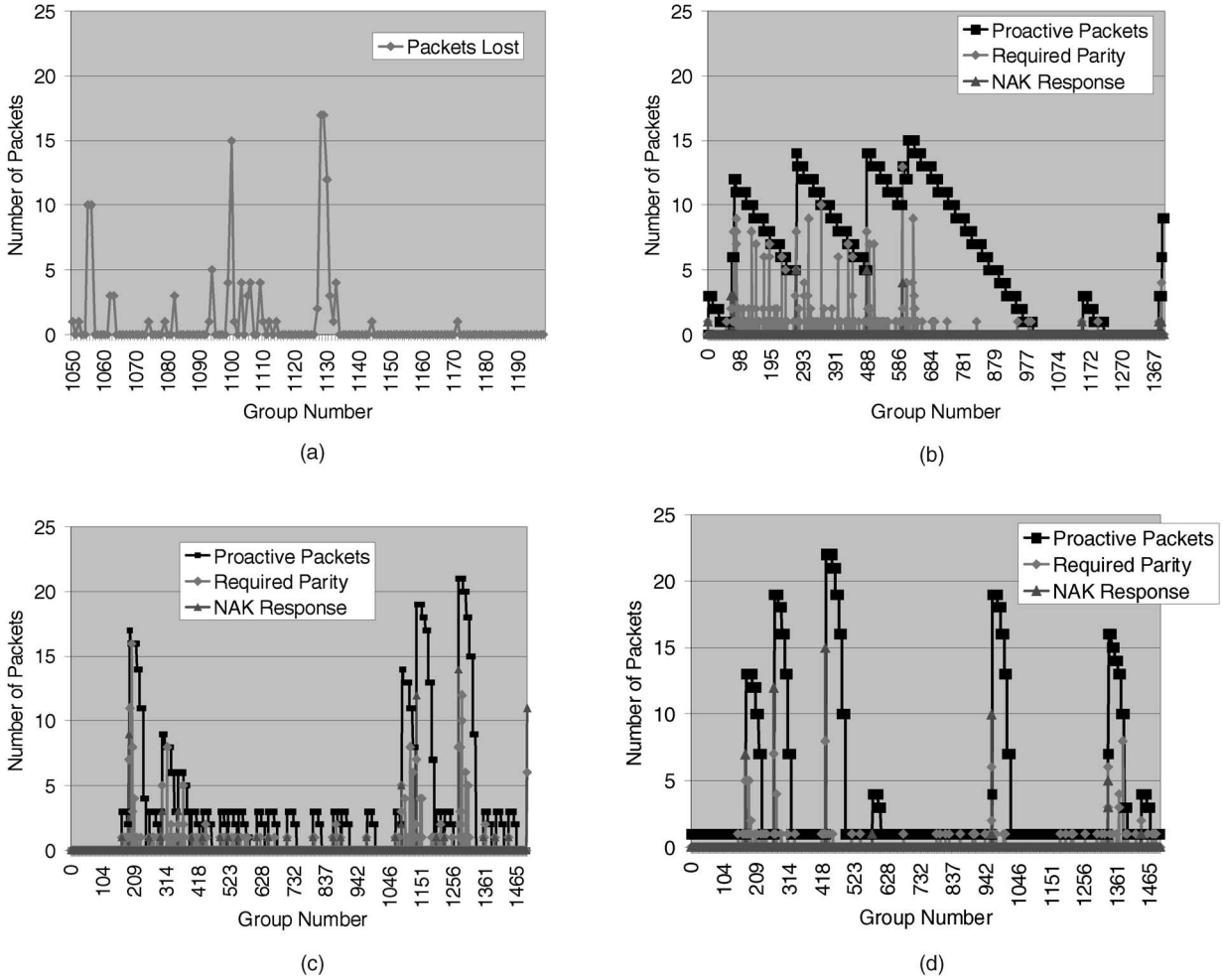

Fig. 12. Experimental results for W-WBRM in mobile testbed, where errors are bursty. (a) Sample trace of measured packe loss. (b) $\alpha_{\mathrm{dec}} = 0.02$. (c) $\alpha_{\mathrm{dec}} = 2^i(0.02)$, lower bound 0 parity packets. (d) $\alpha_{\mathrm{dec}} = 2^i(0.02)$, lower bound 1 parity packet.

larger numbers of receivers. Other than the number of nodes, the simulated environment is similar to the experimental testbed, comprising an Ethernet domain and a WaveLAN domain connected via a wireless access point. The sender node and proxy node, with 500MHz processors, are located in the Ethernet domain. Up to 25 wireless nodes, each with a 300 MHz processor, are located in the WaveLAN domain. The channel bandwidth is 100Mbps for Ethernet domain and 2Mbps for WaveLAN domain.

Local NAK Suppression. The W-WBRM protocol uses local NAK suppression (LNS), whereby a receiver that has sent a NAK for group G will not send another NAK for G until a minimum period of time has elapsed. This technique gives the proxy a fair chance to respond to the NAK, reducing the number of redundant NAKs. The first issue addressed through simulation concerns this local NAK suppression timer. Even though the proxy may respond immediately to a NAK by sending additional parity packets, those packets may be buffered for considerable time at the access point before being transmitted on the wireless channel. If the LNS value is too small, then a repeated NAK will be generated by the receiver before the proxy has a chance to respond. To address this problem, the LNS timer was increased from its initial value of 10 msec to 100 msec. The disadvantage of doing so is that a lost NAK or lost retransmission will not be responded to in a timely manner, however, in all simulation runs, this modification reduced the number of NAKs significantly.

Global NAK Suppression. To further reduce the probability of NAK implosion at the proxy, the simulation study also assessed the merits of *global* NAK suppression (GNS) [14] in the wireless environment. In this approach, each receiver listens for NAKs from other receivers and cancels any pending NAKs that are subsumed by other NAKs. Implementing GNS in W-WBRM required two modifications to the protocol. First, all NAKs are multicast instead of unicast. Second, when a node determines that it needs more parity packets for a given group, it sets a timer to a random value between 0 and $\mathrm{GNS}_{\mathrm{MAX}}$ milliseconds. If the node observes a NAK that subsumes its own NAK before the timer fires, then the node cancels the NAK request. Multicasting NAKs provides little advantage for small numbers of receivers. However, for larger numbers of receivers, the simulations showed that combining GNS and LNS improves throughput by approximately 30 percent. Fig. 13a summarizes the results of adding GNS to W-WBRM. Perhaps the most important observation is that the throughput is relatively constant from 1 to 25 receivers, indicating that the algorithm scales reasonably well, even in the presence of 20 percent packet losses.

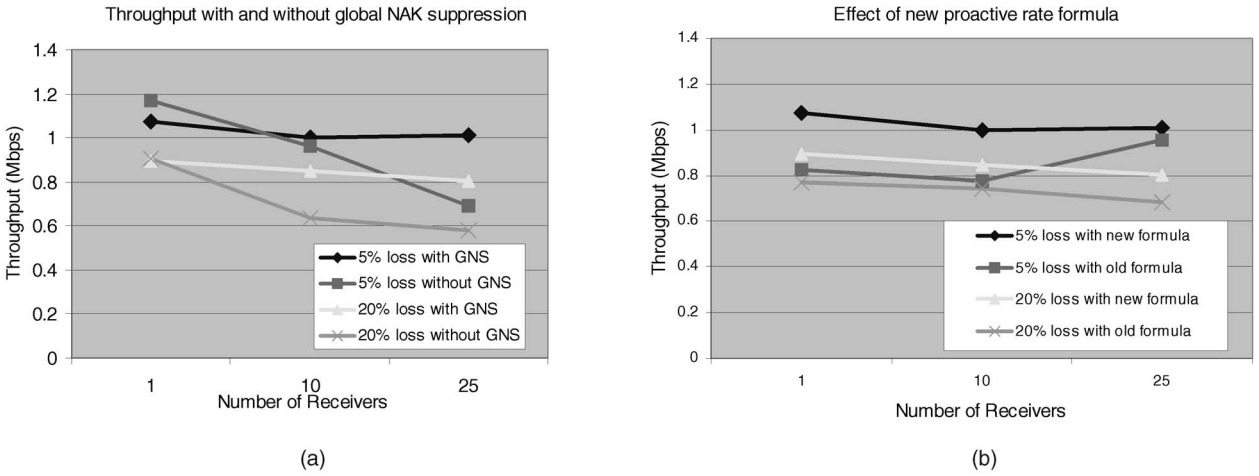

Fig. 13. Effects on throughput of GNS and α_{inc}. (a) With and without GNS. (b) Old α formula versus new α formula.

Increasing the Proactive Rate. Finally, even though the above methods reduced the number of NAKs, the simulations revealed that the value of α sometimes grows to be large due to NAKs from *different* groups arriving at the proxy at approximately the same time. Since GNS operates on a per-group basis (a NAK is suppressed only by a NAK associated with the same group), uncorrelated losses can still yield active NAKs for different groups. Since they are from different groups, none will cancel the other (parity packet suppression) and the proxy will respond to all of them. Using the original formula ($\alpha = \alpha + \alpha_{\text{inc}}$) causes α to increase quickly under these conditions. To address this problem, α is adjusted based on the number of parity packets already being sent in the same group, referred to as

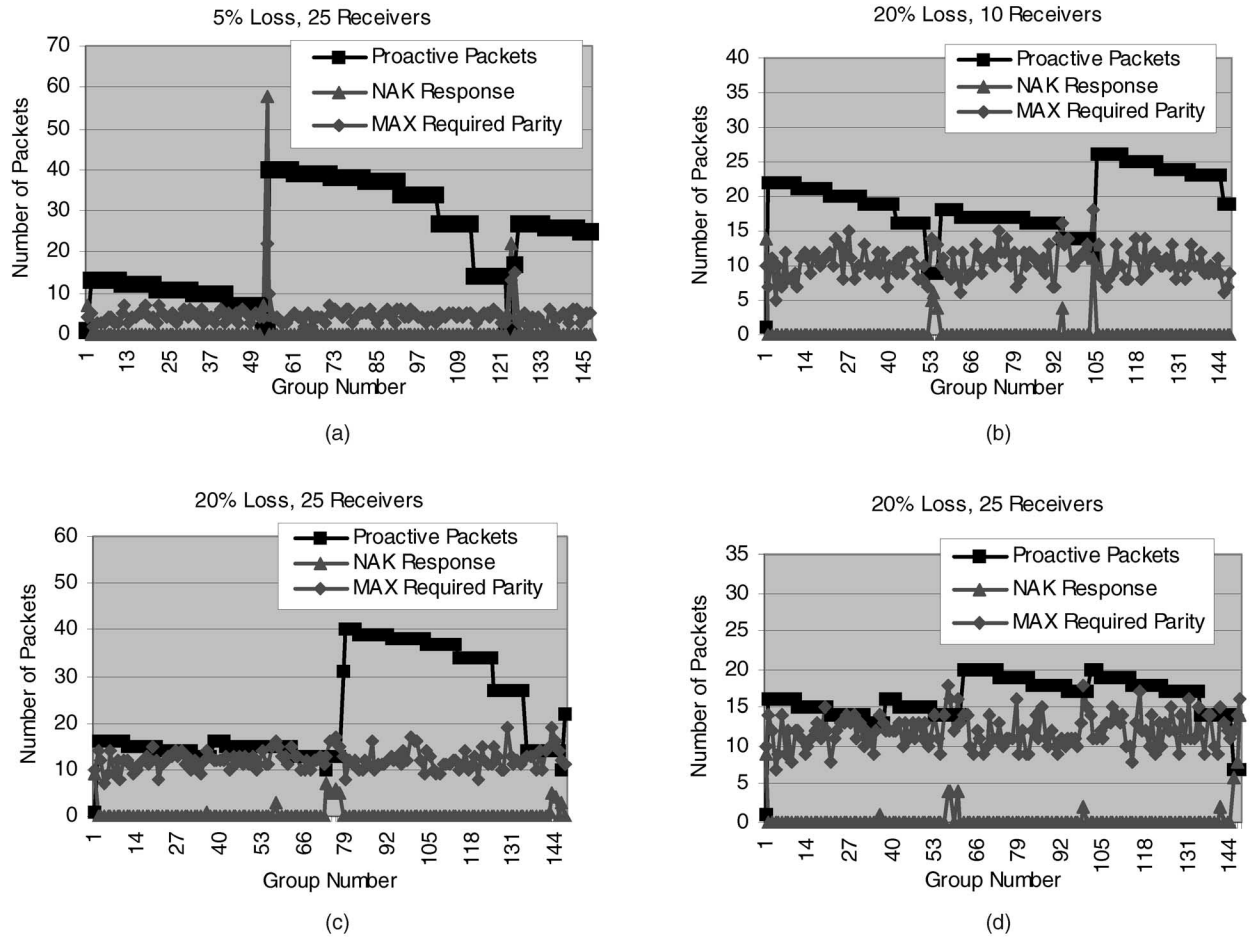

Fig. 14. Traces showing effects of GNS, α_{inc} function, and LNS timer value. (a) New α formula, GNS off, LNS = 100 ms. (b) New α formula, GNS on, LNS = 10 ms. (c) Old α formula, GNS on, LNS = 100 ms. (d) New α formula, GNS on, LNS = 100 ms.

$P(G)$. The resulting formula is $\alpha = \max(\alpha, P(G)/k + \alpha_{\text{inc}})$. As shown in Fig. 13b, this approach improves performance by 10-15 percent under conditions of high loss.

Sample Traces. Fig. 14 shows four sample traces that illustrate the issues discussed above. Fig. 14a shows a sample trace file for 5 percent losses on 25 receivers, using the new α formula and an LNS timer value of 100 msec, but without GNS. Fig. 14b shows a sample trace file for 20 percent losses on 10 receivers, using the new α formula with GNS, with the LNS timer set to only 10 milliseconds. Fig. 14c shows a sample trace file for 20 percent losses on 25 receivers, using the old α formula but with GNS, and an LNS timer of 100 milliseconds. In all three cases, the proactive rate can increase very quickly and remain relatively high. These unneeded parity packets delay the transmission of data packets in subsequent groups, and throughput suffers. Fig. 14d shows a sample trace file for 20 percent losses on 25 receivers, using the new α formula and GNS, with the LNS timer set to 100 msec. As shown, the proactive rate does not exhibit the erroneous behavior of the earlier combinations. Indeed, this combination resulted in the best overall performance.

7 Related Work

In recent years, numerous groups have addressed the problem of reliable data delivery over wireless networks. Areas of study related to the work presented here include error control for wireless *unicast* connections [15], [16], [17], [18] and link-layer support for reliable multicasting [19], [20], [21], [22]. In this section, we focus on user-level FEC-based reliable protocols that do not require new link-layer support. Although several such protocols have been proposed [23], [3], [24], [25], [26], we discuss three that are most closely related to W-WBRM.

First, the RMDP protocol proposed by Rizzo and Vicisano [3] is an FEC-based reliable multicast protocol to be used over the MBone and wireless mobile networks with asymmetric communication channels. RMDP is a hybrid FEC+ARQ protocol that uses several operating parameters that are set according to the type of network. One such parameter, D, the *expansion factor*, is the rate at which parity packets are sent unconditionally with the data packets. The protocol uses global NAK suppression by multicasting NAKs and staggering their transmission randomly, as in SRM [27]. The α parameter in the W-WBRM protocol is similar to D in RMDP, except that α is applied to all transmissions, including sets of of parity packets sent in response to NAKs. In RMDP, apparently D is fixed for a given environment, but the authors do discuss adaptability in terms of changing the value of n at the encoder. By allowing the parameter α to adapt to loss conditions, the W-WBRM protocol quickly tunes the level of redundancy to match the needs of receivers.

Nonnenmacher et al. [24] present an extensive analysis of the relative merits of using an integrated FEC+ARQ protocol, compared to using a separate FEC layer beneath an ARQ-based protocol. The authors describe an integrated FEC-based multicast protocol, NP, for multicast data delivery in the Internet. The NP protocol tries to keep the number of packets transmitted to a minimum at the expense of latency, and does not send more parity packets than requested. However, the protocol uses pipelining of groups to improve throughput: the sender transmits data packets of group $i + 1$ while waiting for NAK(s) for group i. NAKs are multicast, and SRM-like global NAK suppression is used to reduce feedback from receivers. The approach in W-WBRM protocol is more aggressive than that of NP, with a goal of achieving low latency for relatively small resources and good throughput for large ones. Like NP, W-WBRM pipelines the transmission of groups, but uses proactive packets on both data and parity-only transmissions, instead of global NAK suppression alone.

Gemmell et al. [25] describe two FEC-based reliable multicast protocols to be used in one-to-many tele-presentations over the Internet. One of these, FCAST, is intended for bulk transfer of session-persistent data and does not involve sender feedback, but rather uses an FEC-based *carousel*. The other protocol, ECSRM, is closer in design to W-WBRM and is intended for multicasting dynamic data during a session. The ECSRM protocol uses both FEC and global NAK suppression. The parity packet suppression method of W-WBRM, used to avoid sending redundant parity packets in response to multiple NAKs, is similar to the method used in ECSRM. However, the ECSRM protocol does not use proactive transmission of parity packets, and does not adapt to changing loss conditions in the network. Since ECSRM is designed for large groups on the Internet, with long round-trip delays, such adaptation may not be effective. W-WBRM, operating on a proxy in a wireless LAN environment, can make better use of a proactive adaptive mechanism.

8 Conclusions and Future Work

This paper has described a study in the use of proxy services to support Web-based collaboration when some of the participants are located on wireless LANs. An FEC proxy was constructed by extending the WBRM protocol to include adaptive FEC. Experiments and simulations showed that a proactive approach to sending parity packets can reduce feedback by dynamically adapting the rate of redundancy in response to changes in packet loss rate. Topics of our ongoing and future work include: additional analysis and experimentation of various W-WBRM parameters, including α_{inc} and α_{dec}; simulation studies to determine parameter settings for faster wireless networks and larger numbers of receivers; investigation of methods to differentiate queueing losses from propagation losses; and a performance study of the W-WBRM protocol as used in Pocket Pavilion, a collaborative application for wireless handheld computers.

Further Information. A number of related papers and technical reports of the Software Engineering and Network Systems Laboratory can be found at http://www.cse.msu.edu/sens.

Acknowledgments

The authors would like to thank Robel Barrios, Jesus Arango, Aaron Malenfant, Ji Li, Udiyan Padmanabhan, Peng Ge, and Nandagopal Ancha for their contributions to this work. This work was supported in part by the US Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744. This work was also supported in part by US National Science Foundation grants CDA-

9617310, NCR-9706285, CCR-9912407, EIA-0000433, and EIA-0130724. A concise and preliminary version of this paper appeared in the *Proceedings of the IEEE Symposium on Applications and the Internet* (SAINT 2001), San Diego, California, January 2001. This work was conducted while A.P. Mani was a graduate student at Michigan State University.

REFERENCES

[1] P.K. McKinley, A.M. Malenfant, and J.M. Arango, "Pavilion: A Distributed Middleware Framework for Collaborative Web-Based Applications," *Proc. ACM SIGGROUP Conf. Supporting Group Work,* pp. 179-188, Nov. 1999.

[2] A.J. McAuley, "Reliable Broadband Communications Using Burst Erasure Correcting Code," *Proc. ACM SIGCOMM,* pp. 287-306, Sept. 1990.

[3] L. Rizzo and L. Vicisano, "RMDP: An FEC-Based Reliable Multicast Protocol for Wireless Environments," *ACM Mobile Computer and Comm. Rev.,* vol. 2, Apr. 1998.

[4] P.K. McKinley, U.I. Padmanabhan, and N. Ancha, "Experiments in Composing Proxy Audio Services for Mobile Users," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware 2001),* pp. 99-120, Nov. 2001.

[5] J. Arango and P.K. McKinley, "VGuide: Design and Performance Evaluation of a Synchronous Collaborative Virtual Reality Application," *Proc. IEEE Int'l Conf. Multimedia and Expo,* July 2000.

[6] P.K. McKinley, R.R. Barrios, and A.M. Malenfant, "Design and Performance Evaluation of a Java-Based Multicast Browser Tool," *Proc. 19th Int'l Conf. Distributed Computing Systems,* pp. 314-322, 1999.

[7] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," *IEEE J. Selected Areas in Comm.,* vol. 15, no. 3, pp. 398-406, Apr. 1997.

[8] A. Koifman and S. Zabele, "RAMP: A Reliable Adaptive Multicast Protocol," *Proc. IEEE INFOCOM,* pp. 1442-1451, Mar. 1996.

[9] P.K. McKinley and A.P. Mani, "An Experimental Study of Adaptive Forward Error Correction for Wireless Collaborative Computing," *Proc. IEEE 2001 Symp. Applications and the Internet (SAINT-01),* pp. 157-166, Jan. 2001.

[10] R. O'Hara and A. Petrick, *IEEE 802.11 Handbook : A Designer's Companion.* IEEE, 1999.

[11] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM Computer Comm. Rev.,* Apr. 1997.

[12] S. Liang, *Java*™ Native Interface: Programmer's Guide and Specification. Addison-Wesley, 1999.

[13] C. Tang and P.K. McKinley, "MX: A Tool for Emulation and Simulation of Distributed Applications and Protocols," Technical Report MSU-CSE-01-19, Computer Science and Eng., Michigan State Univ., East Lansing, June 2001.

[14] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking,* Dec. 1997.

[15] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Networking,* Dec. 1997.

[16] A. Bakre and B. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE/ACM Trans. Networking,* vol. 46, no. 4, 1997.

[17] D.A. Eckhardt and P. Steenkiste, "A Trace-Based Evaluation of Adaptive Error Correction for a Wireless Local Area Network," *Mobile Networks and Applications,* vol. 4, no. 4, pp. 273-287, 1999.

[18] C. Parsa and J.J. Garcia-Luna-Aceves, "TULIP: A Link-Level Protocol for Improving TCP over Wireless Links," *Proc. IEEE Wireless Comm. and Networking Conf.,* pp. 21-24, Sept. 1999.

[19] J. Kuri, S. Kasera, "Reliable Multicast in Multi-Access Wireless LANs," *Proc. INFOCOM '99,* Mar. 1999.

[20] K. Tang and M. Gerla, "MAC Layer Broadcast Support in 802.11 Wireless Networks," *Proc. IEEE MILCOM,* Oct. 2000.

[21] K. Tang and M. Gerla, "Random Access MAC for Efficient Broadcast Support in Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* Sept. 2000.

[22] Y. Xu and T. Zhang, "An Adaptive Redundancy Technique for Wireless Indoor Multicasting," *Proc. Fifth IEEE Symp. Computers and Comm.,* July 2000.

[23] C. Huitema, "The Case for Packet Level FEC," *Proc. IFIP 5th Int'l Workshop Protocols for High-Speed Networks (PfHSN '96),* pp. 110-120, Oct. 1996.

[24] J. Nonnenmacher, E.W. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Trans. Networking,* vol. 6, no. 4, pp. 349-361, 1998.

[25] J. Gemmell, E. Schooler, and R. Kermode, "A Scalable Multicast Architecture for One-to-Many Telepresentations," *Proc. IEEE Int'l Conf. Multimedia Computing Systems,* pp. 128-139, 1998.

[26] R. Kermode, "Scoped Hybrid Automatic Repeat ReQuest with Forward Error Correction (SHARQFEC)," *Proc. ACM SIGCOMM,* Sept. 1998.

[27] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Applications Level Framing," *Proc. SIGCOMM '95,* pp. 342-356, 1995.

Philip K. McKinley received the BS degree in mathematics and computer science from Iowa State University in 1982, the MS degree in computer science from Purdue University in 1983, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1989. He is currently a professor in the Department of Computer Science and Engineering at Michigan State University. He was previously a member of technical staff at Bell Laboratories. Dr. McKinley is an associate editor for *IEEE Transactions on Parallel and Distributed Systems* and is cochair of the program committee for IEEE International Conference on Distributed Computing Systems 2003. His current research interests include adaptive middleware, collaborative applications, mobile computing, and group communication protocols. He is a member of the IEEE and the IEEE Computer Society.

Chiping Tang received the BS degree from Peking University, Beijing, China, in 1992, and the MS degree from Michigan State University in 2002, both in computer science. Currently, he is a PhD student in the Department of Computer Science and Engineering at Michigan State University. He worked as a software engineer for CS&S in Beijing from 1992 to 1996 and for NUS in Tokyo from 1996 to 1999. His current research interests include adaptive group communication, wireless networking, and computer simulation.

Arun Mani received the MS degree in computer science from Michigan State University in 2000. Currently, he is a member of technical staff at Bell Labs, Lucent Technologies, in Holmdel, New Jersey. His current interests include theory and design of communication systems, analyzing behaviors of combinatorial optimization problems, and study and design of adaptive learning algorithms.